An Adaptive Resolution Tree Visualization of Large Influenza Virus Sequence Datasets

Leonid Zaslavsky, Yiming Bao, and Tatiana A. Tatusova

National Center for Biotechnology Information,
National Library of Medicine, National Institutes of Health,
8600 Rockville Pike, Bethesda, MD 20894, USA
{zaslavsk,bao,tatiana}@ncbi.nlm.nih.gov
http://www.ncbi.nlm.nih.gov

Abstract. Rapid growth of the amount of influenza genome sequence data requires enhancing exploratory analysis tools. Results of the preliminary analysis should be represented in an easy-to-comprehend form and allow convenient manipulation of the data.

We developed an adaptive approach to visualization of large sequence datasets on the web. A dataset is presented in an aggregated tree form with special representation of sub-scale details. The representation is calculated from the full phylogenetic tree and the amount of available screen space. Metadata, such as distribution over seasons or geographic locations, are aggregated/refined consistently with the tree. The user can interactively request further refinement or aggregation for different parts of the tree.

The technique is implemented in Javascript on client site. It is a part of the new AJAX-based implementation of the NCBI Influenza Virus Resource.

Keywords: visualization, adaptive, sequence, tree, phylogenetic, virus, influenza, JavaScript, AJAX.

1 Introduction

The number of influenza virus sequences in the public database more than doubled from the beginning of 2005, thanks to collaborative genome sequencing efforts by the National Institute of Allergy and Infectious Diseases ([1], [2]), St. Jude Children's Research Hospital, the Centers for Disease Control and Prevention, and many others. This requires more sophisticated preliminary analysis tools to be provided to users. Datasets should be represented in an easily comprehendible and adjustable visual form that provides a convenient way of manipulating the data.

The visualization approaches used in several releases of the NCBI Influenza Virus Resource ([3],[4]) were based on sequence-level representation of the data. They provided a convenient interface for viewing the entire dataset and manipulating individual sequences: viewing multiple sequence alignments and trees built

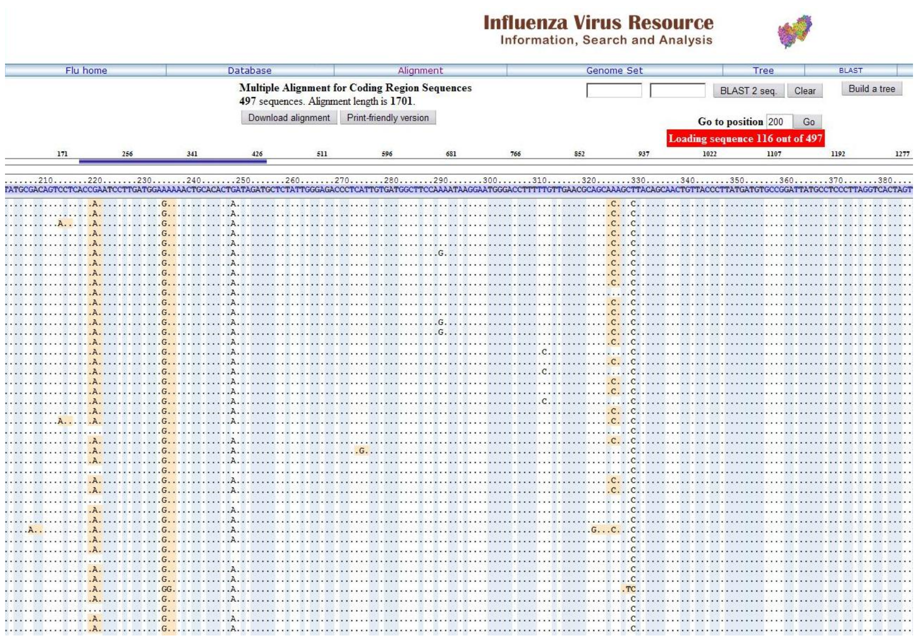

Fig. 1. Multiple sequence alignment

using different algorithms [5] (typical web visualizations of a multiple sequence alignment and a phylogenetic tree are shown in Figures 1, 2). However, the approach based on manipulating individual sequences is not very useful for large datasets. For example, detailed schematic representation of a huge dataset with a fine level of detail, with all information included regardless of relevance, is very difficult to comprehend ([6], [7]). There are many influenza virus sequences that are identical or highly similar to each other. Most of the time, it is not necessary to show all such sequences in the analysis. Also, operating the data manually sequence-by-sequence is highly inefficient and time-consuming for the user. In addition, the user needs guidance to scan through a complex set of data provided not only at the level of individual sequences but also groups of sequences, depending on the task. It is preferable to structure the dataset and provide meaningful aggregated representations with the ability to adapt the aggregation level.

Several systems have been developed to support interactive browsing of large trees with the ability to focus ([8], [9], [10], [11]). The issues of scalability, performance and robustness of tree visualization have been also addressed [12]. In addition, innovative approaches to visualization of geographic information have been developed [13]. Modern cartographical systems widely used in mobile devices provide adaptively-coarsened visual representations of maps. Such representation changes in real time to provide the best visualization suiting a specific task (driving, flying). In each of these cases, the information helpful for performing

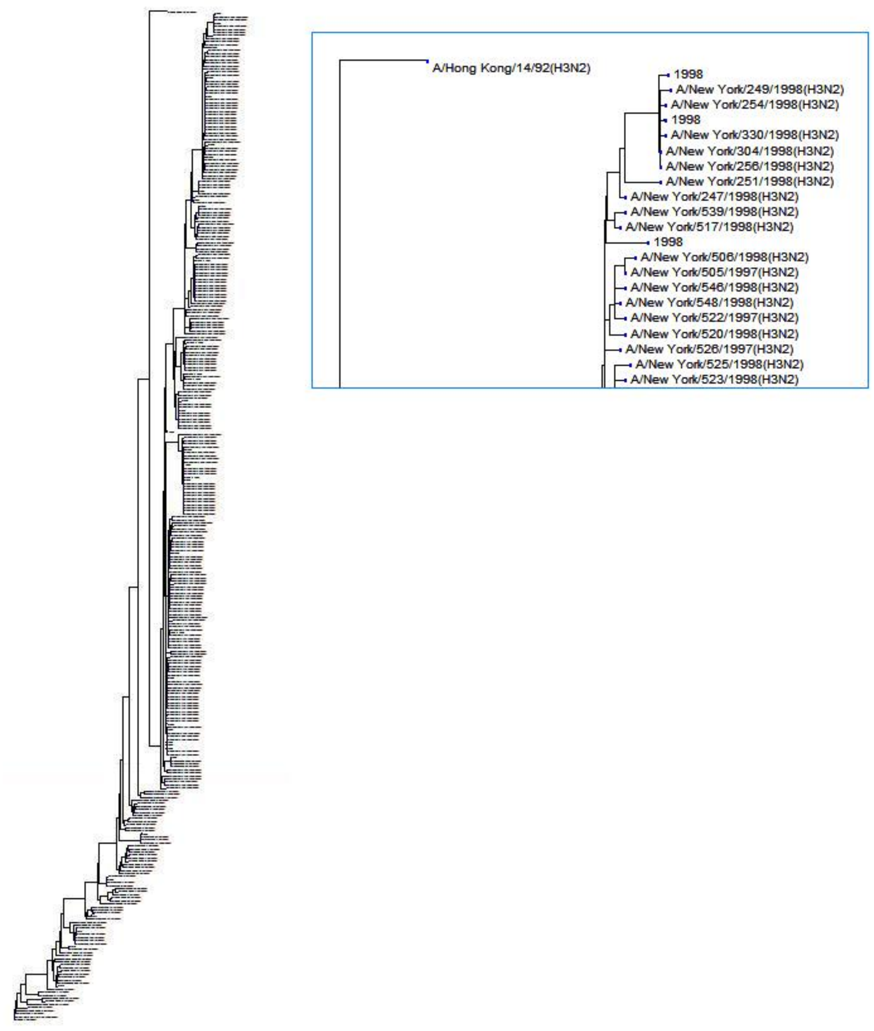
A/Hong Kong/14/92(H3N2)
1998
A/New York/249/1998(H3N2)
A/New York/254/1998(H3N2)
1998
A/New York/330/1998(H3N2)
A/New York/304/1998(H3N2)
A/New York/256/1998(H3N2)
A/New York/251/1998(H3N2)
A/New York/247/1998(H3N2)
A/New York/539/1998(H3N2)
A/New York/517/1998(H3N2)
1998
A/New York/506/1998(H3N2)
A/New York/505/1997(H3N2)
A/New York/546/1998(H3N2)
A/New York/548/1998(H3N2)
A/New York/522/1997(H3N2)
A/New York/520/1998(H3N2)
A/New York/526/1997(H3N2)
A/New York/525/1998(H3N2)
A/New York/523/1998(H3N2)

Fig. 2. An full-resolution tree built for 380 HA protein sequences for Influenza A H3N2 viruses extracted from human hosts during a 20-year period (1968-1998), using the neighbor-joining method. The top of the tree is enlarged in the small window.

the task is provided. The knowledge is represented in an easy-to-comprehend form and the amount of information is limited in a way that a human (driver) can process it and make a reasonable decision in real time.

We propose an adaptive approach to visualize the dataset in an aggregated form adapted to the user's screen, allowing the user to interactively refine or aggregate visualization of different parts of the dataset, depending on the task and need for details. The essential parts of our technique are:

– Representation of a large tree by a smaller tree having aggregated groups as terminal nodes;
– Placing a specially constructed tree to show the structure of each aggregated group at the sub-scale resolution level.
– Creating metadata description for each aggregated group from the original metadata.

Sub-scale resolution representation. When a tree for aggregated group is calculated, it can be shown as a phylogenetic tree with groups shown as named terminal nodes. However, this representation can be refined within the same screen space. Since the height of the font used in annotation is usually several pixels (typically, 10-12 px), available vertical space can be used to show, in some form, the structure of the subtree corresponding to the aggregated group.

An example of aggregated tree is shown in Figure 3 (the dataset is the same as in Figure 2).

Initial tree visualization, built from the full tree taking into account available screen space, can be changed by the user interactively through requesting refinement for some aggregated groups and further aggregating subtrees that are not of interest to the user. The user can also change the tree annotation

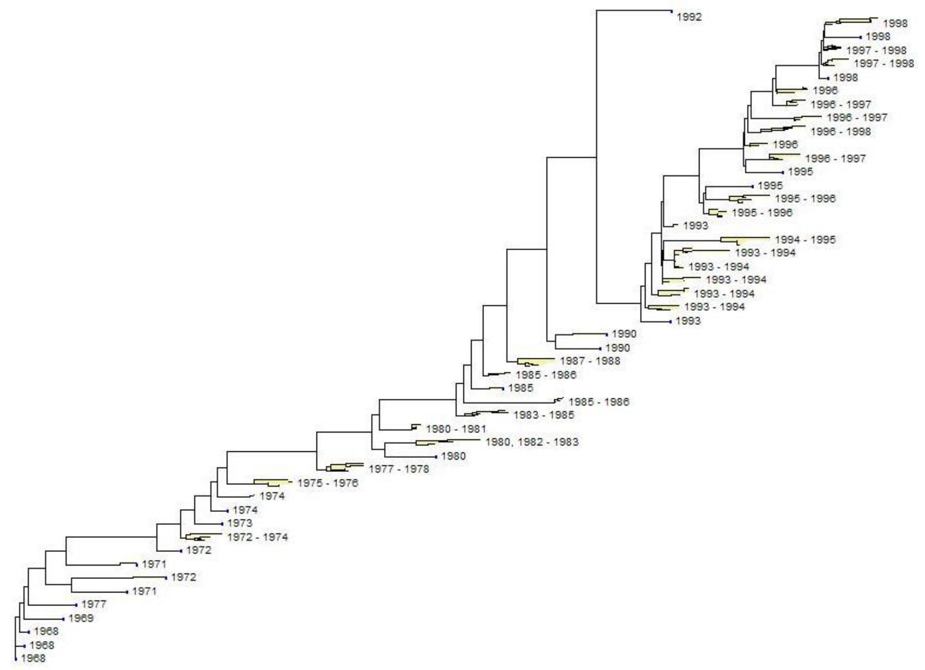

Fig. 3. An aggregated tree built for 380 HA protein sequences for Influenza A H3N2 viruses extracted from human hosts during 20-year period 1968-1998 (the full tree was calculated using the neighbor-joining method). The dataset is the same as in Figure 2.

by indicating special interest for particular influenza, subtypes, years, seasons or geographic locations. This will cause sequences of interest to be recolored at the sequence-level representation, and annotation of aggregated groups to be changed and recolored, to reflect information requested by the user.

While full tree is built on the server, its adaptive visualization is built and can be interactively changed on the client-machine using a JavaScript implementation. It is embedded in the new version of our analysis tools based on the AJAX technology [14]. Since we specifically aim work within a browser on client machine, we are limited to graphic functionality available in HTML[1].

2 Methods

Data Structures. The tree is implemented using an array of nodes, with each node containing array indexes of its parent node and children. We defined the following Javascript objects:

`Tree` - object containing an array of `Node` objects as its `nodes` property, root index in the node array as its `rootId` property, and some auxiliary objects. In the process of adaptive aggregation, object`SubtreeInfo` is added.

`Node` - an object having the following properties:

`parentId`	- an index of the parent node in the array `nodes`, or `-1` for root;
`children`	- an array containing indices of children nodes in `nodes` array;
`branchLength`	- length of the branch going to the parent node;
`metadata`	- node metadata (name, subtype, date of extraction, country, etc.);
`presentation`	- information on visual representation;
`status`	- an integer value, initially equal to `0`. Value `1` is set of of corresponding subtree should be represented in the aggregated form.

Object `SubtreeInfo` has the following fields:

`lengthMin`	- minimal distance from the root of the subtree to a leaf;
`lengthMax`	- maximal distance from a leaf of the subtree to a leaf;
`diam`	- diameter of the subtree.

Distances used in calculations of `lengthMin`, `lengthMin`, and `lengthMin` are tree distances, i.e. lengths of shortest paths.

Calculating subtree information for original tree. `SubtreeInfo` objects, containing minimal and maximal distances from the subtree root to a leaf and

[1] Non-linear two-dimensional transformations requiring rich graphical functionality are the core of the approaches like [10].

diameter of the subtree are calculated for tree nodes in a bottom-to-top manner using formulas:

$$l_i^{min} = \min\{l_j + l_j^{min} | j \in \Omega_i\}; \tag{1}$$

$$l_i^{max} = \max\{l_j + l_j^{max} | j \in \Omega_i\}; \tag{2}$$

$$d_i = \max\left(\max\{d_j | j \in \Omega_i\}, \max\{d_j + d_k + l_j + j_k | j, k \in \Omega_i, j \neq k\}\right); \tag{3}$$

where

Ω_i	is the subtree having node i as its root,
l_i	is the branch length from node i to its parent,
l_i^{min}	is minimal tree distance from node i to a leaf in subtree Ω_i,
l_i^{max}	is maximal tree distance from node i to a leaf in subtree Ω_i,
d_i	is diameter of subtree Ω_i.

Building an aggregated tree. In order to control visualization, integer *status* variable `Tree.nodes[i].status` is assigned to each node i of the full tree. It has the following meaning:

If $status = 0$, the node is treated as a usual tree node.
If $status = 1$, the node is treated as an aggregated group:
- The name of the aggregated group is created and shown;
- The aggregated group is shown graphically using sub-scale visualization.

In the beginning, we assign root *status* to *1*, i.e. consider the tree as aggregated in one group. We start disaggregate nodes, from the child of the root that has largest diameter of the corresponding subtree, and disaggregate while screen space allows (Technically, desegregating node i means setting its *status* to *0* and setting *status* of its children to *1*).

To control the order of node disaggregation, we use auxiliary array Θ, where indices of the candidate nodes for disaggregation sorted by non-increasing diameters are placed:

$$d_{i_k} \geq d_{i_m} \text{ for any } k < m, 0 \leq k, m < |\Theta| \tag{4}$$

Technically, array Θ is included in the `Tree` object as `Tree.leafArray.nodeIds`. It is easy to see that

$$d_{i_0} = \max\{d_{i_k} \mid 0 \leq k < |\Theta|\} \tag{5}$$

i.e., the node in the front of the array has the maximal diameter of the subtree.

The disaggregation algorithm is described as follows. The number of groups in the aggregated tree is denoted as N, the maximal allowed number of groups as N_{max}, and the set of children of node i as Λ_i.

Algorithm 1.

```
Set root $status$ to 1;
Include root in $\Theta$;
Set $N$ to 1.
While( $|\Theta| > 0$ and $N + \max(|\Lambda_{i_0}| - 1,\ 0) \leq N_{max}$){
   Set $status$ of node $i_0$ to $0$;
   Delete $i_0$ from $\Theta$;
   If( $\Lambda_{i_0} \neq \emptyset$ ){
      For ( all $k \in \Lambda_{i_0}$ ){
         Include $k$ in $\Theta$;
         Set $status$ of node $k$ to $1$;
      }
      Set $N \leftarrow N + |\Lambda_{i_0}| - 1$.
   }
}
```

The new indices k are included in Θ with order preserved[2] (4).

Drawing sub-scale resolution representation. We represent a subtree on sub-resolution level (e.g., in the space approximately equal to the font height) as follows:

- Start from a tree containing only one element, corresponding to the root of the sub-tree and perform several steps of disaggregation;
- Represent each non-resolved subtree by two leaves: closest to the root and most distant (see Fig. 4).

Figure 5 illustrates transformation of a subtree in its sub-scale resolution representation. The algorithm for building a subscale-resolution tree is similar to Algorithm 1.

Aggregating Metadata. When aggregated groups of sequences are created, we can create abstracted description of the group to annotate the tree. We can summarize the group using the following descriptive characteristics:

- type;
- subtype,
- year of extraction;
- season of extraction;
- geographical location (country, continent).

However, abstracting or summarizing less formal descriptions, such as strain name, seems to be more challenging.

[2] In our current implementation, a binary search is performed to find insertion position and JavaScript method `Array::splice` is used for inserting an element in the JavaScript array.

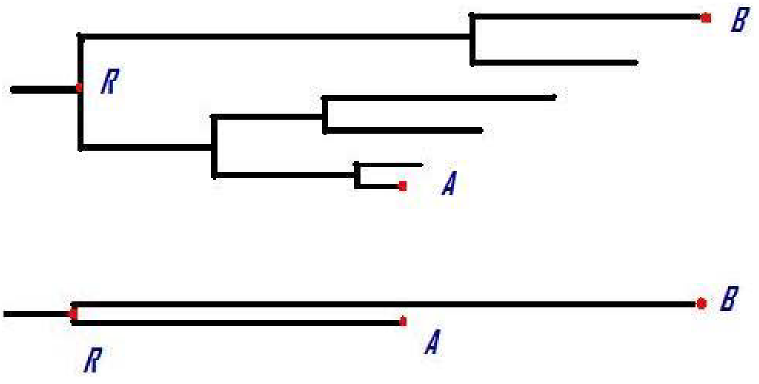

Fig. 4. Representation of of an unresolved subtree by a tree with two leaves, showing the leaf closest to the root and the leaf most distant from the root, in the original subtree

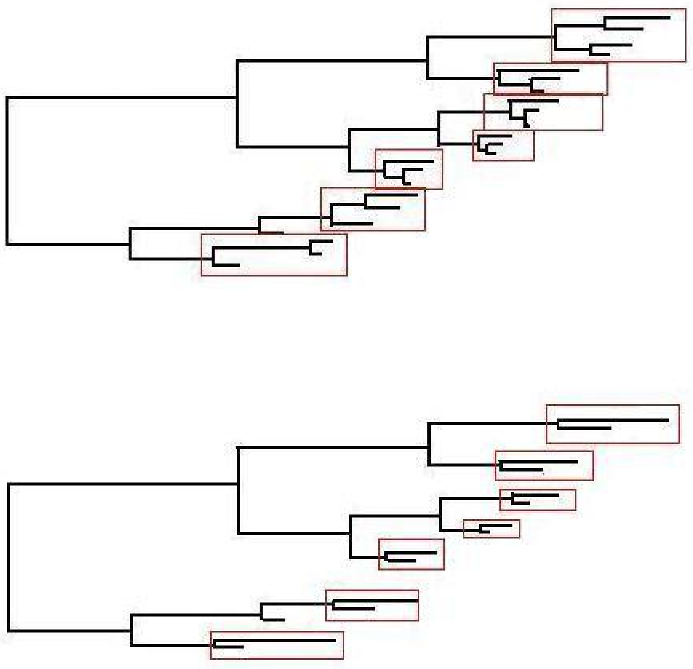

Fig. 5. A subtree (top) and its sub-scale resolution representation (bottom)

JavaScript Implementation. The JavaScript library implementing the adaptive visualization of the tree consists of two layers. The first contains objects and methods to calculate the tree for aggregated groups and trees for sub-level resolution, as well as metadata. The second contains objects and methods for actual rendering; it creates and changes HTML objects and places them to DOM tree. Separation of the logical and rendering levels facilitates easy change of rendering when necessary, the logic and data flow are completely abstracted from rendering technology.

Currently, we implement tree rendering using HTML ⟨`div`⟩ objects. In the future, however, we may decide to take a different approach to web rendering using standard non-propietary tools. One potential candidate is the new HTML element ⟨`canvas`⟩ [15], and another one is Scalable Vector Graphics (SVG) [16]. However, neither ⟨`canvas`⟩ element, nor SVG, have become widely used standard tools providing implementation-independent output yet[3,4].

While implementation of adaptive tree visualization purely within HTML using JavaScript allows manipulation of the tree on the client machine, and this approach has obvious advantages, it also has a drawback: the implementation of the tree using HTML elements does not allow saving the tree as an image and quality of printing depends solely on the browser functionality. In the future, we plan to provide an image for the tree: the selection made by the user within the web tool will be transferred to the server and an image in one of standard formats will be created and sent to the client.

Test Results. We applied developed methodology to typical influenza virus sequence datasets. An aggregated tree obtained for dataset containing 380 HA protein sequences for Influenza A H3N2 viruses extracted from human hosts during 20-year period 1968-1998, is shown in Figure 3 (for comparison, see Figure 2 showing the same dataset with a traditional sequence-level resolution).

3 Discussion

Adaptive aggregative visualization of datasets with the possibility to refine and coarse different parts of the representation interactively on the web is a promising approach to a convenient preliminary analysis of large datasets. It allows the user to view and manipulate the data hierarchically, doing each operation at the appropriate resolution level. We implemented this approach for tree visualization and demonstrated its efficiency and usefulness.

It is highly desirable to apply hierarchical visualization and adaptive resolution to other types of data representation. One of the immediate areas requiring our attention is multiple sequence alignment visualization. While we provide a convenient multiple alignment view in the current system (such as shown in Figure 1), the user would not be able to comprehend data at sequence level for

[3] New HTML5 element ⟨`canvas`⟩ is a part of the proposed HTML5 standard, but it is not yet implemented as a native object in all browsers. Moreover, its current limited implementation in selected browsers does not allow to include text [15].

[4] SVG requires either a native implementation within a browser or a plug-in. Partial native implementations are available in selected browsers [17]. Several plug-in implementations are available. However, Adobe Systems, the provider of Adobe SVG Viewer, stated that they will discontinue support for Adobe SVG Viewer by the end of 2007 [18].

large datasets consisting of hundreds or even thousands of sequences. A different alignment representation, that allows to adjust the resolution and focus, seems to be helpful.

Acknowledgements

This research was supported by the Intramural Research Program of the NIH, National Library of Medicine.

The NCBI Influenza Virus Resource is developed and being continuously improved as a part of a collaborative effort, the Influenza Genome Sequencing Project led by NIAID ([19], [1], [2]). The authors are thankful to David J. Lipman, Yury Voronov, Boris Fedorov, Stacy Ciufo and Sergey Ponomarev for productive discussions.

References

1. Fauci, A.S.: Race against time. Nature **435**(7041) (May 2005) 423–424
2. Ghedin, E., Sengamalay, N.A., Shumway, M., Zaborsky, J., Feldblyum, T., Subbu, V., Spiro, D.J., Sitz, J., Koo, H., Bolotov, P., Dernovoy, D., Tatusova, T., Bao, Y., St George, K., Taylor, J., Lipman, D.J., Fraser, C.M., Taubenberger, J.K., Salzberg, S.L.: Large-scale sequencing of human influenza reveals the dynamic nature of viral genome evolution. Nature **437**(7062) (October 2005) 1162–1166
3. The National Center for Biotechnology Information (NIH/NLM/NCBI): The Influenza Virus Resource. `http://www.ncbi.nlm.nih.gov/genomes/FLU/FLU.html`
4. Bao, Y., Bolotov, P., Dernovoy, D., Kiryutin, B., Zaslavsky, L., Tatusova, T.A., Ostell, J., Lipman, D.J.: NCBI Influenza Virus Resource. Manuscript in preparation. National Center for Biotechnology Information (2006)
5. Felsenstein, J.: Inferring Phylogenies. 1 edn. Cambridge University Press (September 2003)
6. Mather, G.: Foundations of Perception. 1 edn. Psychology Press (January 2006)
7. Baron, J.: Thinking and Deciding. 3 edn. Cambridge University Press (December 2000)
8. Card S. K., N.D.: Degree-of-interest trees: A component of an attention-reactive user interface. In: Proc. Advanced Visual Interfaces (AVI). (2002) 231–245
9. Fekete J.-D., P.C.: Interactive information visualization of a million items. In: Proc. InfoVis. (2002) 117–124
10. Lamping J., Rao R., P.P.: Focus+content technique based on hyperbolic geometry for viewing large hierarchies. In: Proc. CHI'95. (1995) 401–408
11. Rost U., B.B.E.: Treewiz: interactive exploration of huge trees. Bioinformatics **18**(1) (2002) 109–114
12. Beermann, D., Munznerz, T., Humphreysy, G.: Scalable, robust visualization of very large trees. In K.W. Brodlie, D.J. Duke, K.I.J., ed.: EUROGRAPHICS - IEEE VGTC Symposium on Visualization. (2005) 1–8
13. MacEachren, A.M.: How Maps Work: Representation, Visualization, and Design. 2nd revised edn. The Guilford Press (June 2004)
14. Zakas, N.C., McPeak, J., Fawcett, J.: Professional Ajax. 1 edn. Wrox (February 2006)

15. Mozilla Foundation: Resources related to the new HTML5 ⟨canvas⟩ element. http://developer.mozilla.org/en/docs/Category:HTML:Canvas
16. The World Wide Web Consortium (W3C): Scalable vector graphics (svg): Xml graphics for the web. http://www.w3.org/Graphics/SVG/
17. Mozilla Foundation: Mozilla svg project. http://www.mozilla.org/projects/svg/
18. Adobe Systems: Adobe SVG Viewer . http://www.adobe.com/svg/viewer/install/
19. The National Institute of Allergy and Infectious Diseases (NIAID): NIAID Launches Influenza Genome Sequencing Project. Press Release. http://www3.niaid.nih.gov/news/newsreleases/2004/flugenome.htm (November 2004)